30장 Date

- 표준 빌트인 객체인 Data는 날짜와 시간을 위한 메서드를 제공하는 빌트인 객체이면서 생성자 함수다.
- UTC(협정 세계시)는 국제 표준시를 말하며 기술적인 표기에는 UTC가 사용된다.
- KST(한국 표준시)는 UTC에 9시간을 더한 시간이다. 즉 KST는 UTC보다 9시간 빠르다.

30.1 Data 생성자 함수

- Date는 생성자 함수다.
- Date 객체는 내부적으로 날짜와 시간을 나타내는 정수값을 갖는다.
- 이 값은 1970년 1월 1일 00:00:00(UTC)를 기점으로 Date 객체가 나타내는 날짜와 시간까지의 밀리초를 나타낸다.
- Data 생성자 함수로 객체를 생성하는 방법은 다음과 같이 4가지가 있다.

30.1.1 new Date()

- Date 생성자 함수를 인수없이 new 연산자와 함께 호출하면 현재 날짜와 시간을 가지는 Date 객체를 반환한다.
- Date 객체는 내부적으로 날짜와 시간을 나타내는 정수값을 갖지만

  Date 객체를 콘솔에 출력하면 기본적으로 날짜와 시간 정보를 출력한다.

```
new Date(); // -> Mon Nov 27 2023 21:20:51 GMT+0900 (대한민국 표준시)
```

- new 연산자 없이 호출하면 Date 객체를 반환하지 않고 날짜와 시간 정보를 나타내는 **문자열**을 반환한다.

```
Date(); // -> "Mon Nov 27 2023 21:20:51 GMT+0900 (대한민국 표준시)"
```

30.1.2 new Date(miliseconds)

- Date 생성자 함수에 숫자 타입의 밀리초를 인수로 전달하면 1970년 1월 1일 00:00:00(UTC)을 기점으로 인수로 전달된 밀리초만큼 경과한 날짜와 시간을 나타내는 Data 객체를 반환한다.

```
// 한국 표준시 KST는 협정 세계시 UTC에 9시간을 더한 시간이다.
new Date(0); // -> Thu Jan 01 1970 09:00:00 GMT+0900 (대한민국 표준시)

/*
86400000ms는 1day를 의미한다.
1s = 1,000ms
1m = 60s * 1,000ms = 60,000ms
1h = 60m * 60,000ms = 3,600,000ms
1d = 24h * 3,600,000ms = 86,400,000ms
*/
new Date(86400000); // -> Fri Jan 02 1970 09:00:00 GMT+0900 (대한민국 표준시)
```

30.1.3 new Date(dateString)

- Date 생성자 함수에 날짜와 시간을 나타내는 문자열을 인수로 전달하면 지정된 날짜와 시간을 나타내는 Date 객체를 반환한다.
- 이때 인수로 전달한 문자열은 Date.parse 메서드에 의해 해석 가능한 형식이어야 한다.

```
new Date('May 26, 2020 10:00:00');
// -> Tue May 26 2020 10:00:00 GMT+0900 (대한민국 표준시)

new Date('2020/03/26/10:00:00');
// "
```

30.1.4 new Date(year, month[, day, hour, minute, second, millisecond])

- Date 생성자 함수에 연, 월, 일, 시, 분, 초, 밀리초를 의미하는 숫자를 인수로 전달하면 지정된 날짜와 시간을 나타내는 Date 객체를 반환한다.
- 이때 연, 월은 반드시 지정해야한다. 지정하지 않은 옵션 정보는 0 또는 1로 초기화 된다.

- 인수는 다음과 같다.

인수	내용
year	연을 나타내는 1900년 이후의 정수. 0부터 99는 1900부터 1999로 처리된다.
month	월을 나타내는 0 ~ 11까지의 정수(주의: 0부터 시작, 0 = 1월)
day	일을 나타내는 1 ~ 31까지의 정수
hour	시를 나타내는 0 ~ 23까지의 정수
minute	분을 나타내는 0 ~ 59까지의 정수
second	초를 나타내는 0 ~ 59까지의 정수
millisecond	밀리초를 나타내는 0 ~ 999까지의 정수

```
// 월을 나타내는 2는 3월을 의미한다. 2020/3/1/00:00:00:00
new Date(2020, 2);
// -> Sun Mar 01 2020 00:00:00 GMT+0900 (대한민국 표준시)

// 월을 나타내는 2는 3월을 의미한다. 2020/3/26/10:00:00:00
new Date(2020, 2, 26, 10, 00, 00, 0);

// 다음처럼 표현하면 가독성이 훨씬 좋다.
new Date('2020/3/26/10:00:00:00');
```

30.2 Date 메서드

30.2.1 Date.now

- 1970년 1월 1일 00:00:00(UTC) 을 기점으로 현재 시간까지 경과한 밀리초를 숫자로 반환한다.

```
const now = Date.now(); // -> 1593971539112

// Date 생성자 함수에 숫자 타입의 밀리초를 인수로 전달하면 1970년 1월 1일 00:00:00(UTC)을
// 기점으로 인수로 전달된 밀리초만큼 경과한 날짜와 시간을 나타내는 Date 객체를 반환한다.
new Date(now);
```

30.2.2 Date.parse

- 1970년 1월 1일 00:00:00(UTC)을 기점으로 인수로 전달된 지정시간(new Date(dateString)) 까지의 밀리초를 숫자로 반환한다.

```
Date.parse('Jan 2, 1970 00:00:00 UTC'); // 86400000
```

30.2.3 Date.UTC

- 1970년 1월 1일 00:00:00(UTC)을 기점으로 인수로 전달된 지정 시간까지의 밀리초를 숫자로 반환한다.
- Date.UTC 메서드는 new Date(year, month[, day, hour, minute, second, millisecond])와 같은 형식의 인수를 사용해야 한다.

```
Date.UTC(1970, 0, 2); // 86400000
Date.UTC('2023/11/27'); // NaN
```

30.2.4 Date.prototype.getFullyear

- Date 객체의 연도를 나타내는 정수를 반환한다.

```
new Date('2020/07/24').getFullYear(); // -> 2020
```

30.2.5 Date.prototype.setFullyear

- Date 객체에 연도를 나타내는 정수를 설정한다. 연도 이외에 옵션으로 월, 일도 설정할 수 있다.

```
const today = new Date();

// 년도 지정
today.setFullYear(2000);
today.getFullYear(); // -> 2000

// 년도/월/일 지정
today.setFullYear(1900, 0, 1);
today.getFullYear(); // -> 1900
```

30.2.6 Date.prototype.getMonth

- Date 객체의 월을 나타내는 0 ~ 11 정수를 반환한다.
- 1월은 0, 12월은 11이다.

```
new Date('2020/07/24').getMonth(); // -> 6
```

30.2.7 Date.prototype.setMonth

- Date 객체에 월을 나타내는 0 ~ 11의 정수를 설정한다. 월 이외에 옵션으로 일도 설정할 수 있다.

```
const today = new Date();

// 월 지정
today.setMonth(0); // 1월
today.getMonth(); // -> 0

// 월/일 지정
today.setMonth(11, 1); // 12월 1일
today.getMonth(); // -> 11
```

30.2.8 Date.prototype.getDate

- Date 객체의 날짜를 나타내는 정수를 반환한다.

```
new Date('2020/07/24').getDate(); // -> 24
```

30.2.9 Date.prototype.setDate

```
const today = new Date();

// 날짜 지정
today.setDate(1);
today.getDate(); // -> 1
```

30.2.10 Date.prototype.getDay

- 객체의 요일을 나타내는 정수를 반환한다.

요일	반환값
일요일	0
월요일	1
화요일	2
수요일	3
목요일	4
금요일	5
토요일	6

```
new Date('2020/07/24').getDay(); // -> 5
```

30.2.11 Date.prototype.getHours

- Date 객체의 시간을 나타내는 정수를 반환한다.

```
new Date('2020/07/24/12:00').getHours(); // -> 12
```

30.2.12 Date.prototype.setHours

- Date 객체에 시간을 나타내는 정수를 설정한다. 시간 이외에서 옵션으로 분,초, 밀리초도 설정할 수있다.

```
const today = new Date();

// 시간 지정
today.setHours(7);
today.getHours(); // -> 7

// 시간/분/초/밀리초 지정
today.setHours(0, 0, 0, 0); //00:00:00:00
today.getHours(); // -> 0
```

30.2.13 Date.prototype.getMinutes

- Date 객체의 분을 나타내는 정수를 반환한다.

```
new Date('2020/07/24/12:30').getMinutes(); // -> 30
```

30.2.14 Date.prototype.setMinutes

- Date 객체에 분을 나타내는 정수를 설정한다. 분 이외에 옵션으로 초, 밀리초도 설정할 수 있다.

```
const today = new Date();

// 분 지정
today.setMinutes(50);
today.getMinutes(); // -> 50

// 분/초/밀리초 지정
today.setMinutes(5, 10, 999); //HH:05:10:999
today.getMinutes(); // -> 5
```

30.2.15 Date.prototype.getSeconds

- Date 객체의 초를 나타내는 정수를 반환한다.

```
new Date('2020/07/24/12:30:10').getSeconds(); // -> 10
```

30.2.16 Date.prototype.setSeconds

- Date 객체에 초를 나타내는 정수를 설정한다. 초 이외에 옵션으로 밀리초도 설정할 수 있다.

```
const today = new Date();

// 초 지정
today.setSeconds(30);
today.getSeconds(); // -> 30
```

```
// 초/밀리초 지정
today.setSeconds(10, 0); //HH:MM:10:000
today.getSeconds(); // -> 10
```

30.2.17 Date.prototype.getMilliSeconds

- Date 객체의 밀리초를 나타내는 정수를 반환한다.

```
new Date('2020/07/24/12:30:10:150').getMilliSeconds(); // -
> 150
```

30.2.18 Date.prototype.setMilliSeconds

- Date 객체에 밀리초를 나타내는 정수를 설정한다.

```
const today = new Date();

// 밀리초 지정
today.setMilliSeconds(123);
today.getMilliSeconds(); // -> 123
```

30.2.19 Date.prototype.getTime

- 1970년 1월 1일 00:00:00(UTC)를 기점으로 Date 객체의 시간까지 경과된 밀리초를 반환한다.

```
new Date('2020/07/24/12:30').getTime(); // -> 1595561400000
```

30.2.20 Date.prototype.setTime

- Date 객체에 1970년 1월 1일 00:00:00(UTC)를 기점으로 경과된 밀리초를 설정한다.

```
const today = new Date();

// 밀리초 지정
today.setTime(86400000); //86400000은 1day를 나타낸다.
```

```
today.getTime(); // -> Fri Jan 02 1970 09:00:00 GMT+0900
(한국 표준시)
```

30.2.21 Date.prototype.getTimezoneOffset

- UTC와 Date 객체에 지정된 로캘시간과의 차이를 분단위로 반환한다.

```
const today = new Date();

// UTC와 today의 지정 로캘KST와의 차이는 -9시간이다.
today.getTimezoneOffset() / 60; // -9
```

30.2.22 Date.prototype.toDateString

- 사람이 읽을 수 있는 형식의 문자열로 Date 객체의 날짜를 반환한다.

```
const today = new Date('2020/7/24/12:30');

today.toString();     // -> Fri Jul 24 2020 12:30:00 GMT+09
00 (대한민국 표준시)
today.toDateString(); // -> Fri Jul 24 2020
```

30.2.23 Date.prototype.toTimeString

- 사람이 읽을 수 있는 형식으로 Date 객체의 시간을 표현한 문자열을 반환한다.

```
const today = new Date('2020/7/24/12:30');

today.toString();     // -> Fri Jul 24 2020 12:30:00 GMT+09
00 (대한민국 표준시)
today.toTimeString(); // -> 12:30:00 GMT+0900 (대한민국 표준
시)
```

30.2.24 Date.prototype.toISOString

- ISO 8601 형식으로 Date 객체의 날짜와 시간을 표현한 문자열을 반환한다.

```
const today = new Date('2020/7/24/12:30');

today.toString();    // -> Fri Jul 24 2020 12:30:00 GMT+090
0 (대한민국 표준시)
today.toISOString(); // -> 2020-07-24T03:30:00.000Z

today.toISOString().slice(0, 10); // -> 2020-07-24
today.toISOString().slice(0, 10).replace(/-/g, ''); // -> 2
0200724
```

30.2.25 Date.prototype.toLocalString

- 인수로 전달한 로캘을 기준으로 Date 객체의 날짜와 시간을 표현한 문자열을 반환한다.
- 인수를 생략한 경우 브라우저가 동작 중인 시스템의 로캘을 적용한다.

```
const today = new Date('2020/7/24/12:30');

today.toString(); // -> Fri Jul 24 2020 12:30:00 GMT+0900
(대한민국 표준시)
today.toLocaleString(); // -> 2020. 7. 24. 오후 12:30:00
today.toLocaleString('ko-KR'); // -> 2020. 7. 24. 오후 12:3
0:00
today.toLocaleString('en-US'); // -> 7/24/2020, 12:30:00 PM
today.toLocaleString('ja-JP'); // -> 2020/7/24 12:30:00
```

30.2.26 Date.prototype.toLocalTimeString

- 인수로 전달한 로캘을 기준으로 Date 객체의 시간을 쵸현한 문자열을 반환한다.
- 인수를 생략한 경우 브라우저가 동작 중인 시스템의 로캘을 적용한다.

```
const today = new Date('2020/7/24/12:30');

today.toString(); // -> Fri Jul 24 2020 12:30:00 GMT+0900
(대한민국 표준시)
today.toLocalTimeString(); // -> 오후 12:30:00
today.toLocalTimeString('ko-KR'); // -> 오후 12:30:00
```

```
today.toLocalTimeString('en-US'); // -> 12:30:00 PM
today.toLocalTimeString('ja-JP'); // -> 12:30:00
```

30.3 Date를 활용한 시계 예제

- 다음 예제는 현재 날짜와 시간을 초 단위로 반복 출력한다.

```
(function printNow() {
  const today = new Date();


  const dayNames = [
    '(일요일)',
    '(월요일)',
    '(화요일)',
    '(수요일)',
    '(목요일)',
    '(금요일)',
    '(토요일)'
  ];
  // getDay 메서드는 해당 요일(0 ~ 6)을 나타내는 정수를 반환한다.
  const day = dayNames[today.getDay()];


  const year = today.getFullYear();
  const month = today.getMonth() + 1;
  const date = today.getDate();
  let hour = today.getHours();
  let minute = today.getMinutes();
  let second = today.getSeconds();
  const ampm = hour >= 12 ? 'PM' : 'AM';


  // 12시간제로 변경
  hour %= 12;
  hour = hour || 12; // hour가 0이면 12를 재할당
```

```
  // 10미만인 분과 초를 2자리로 변경
  minute = minute < 10 ? '0' + minute : minute;
  second = second < 10 ? '0' + second : second;


  const now = `${year}년 ${month}월 ${date}일 ${day} ${hou
r}:${minute}:${second} ${ampm}`;


  console.log(now);


  // 1초마다 printNow 함수를 재귀 호출한다. 41.2.1절 "setTimeout
/ clearTimeout" 참고
  setTimeout(printNow, 1000);
}());
```